

実験計画法の並列分散処理による ブルウィップ効果における 要因の部分効用の解明

情報基盤工学講座 横井 稜

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

March 27, 2020

1.1. 本研究の背景

2/18

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

サプライチェーンの課題の一つに、取引する人の心的要因やサプライチェーンの上流から下流への情報伝達の遅れなどの要因が重なり、サプライチェーンの上流に行くほど需要量のばらつきが増大する現象がある。その現象をブルウィップ効果という。変動に対応するために上流のサプライヤーほど在庫を多く保有する傾向にあり、余剰在庫が経営状態を悪化させる。先行研究では、ブルウィップ効果に影響する要因を限定して考察している。

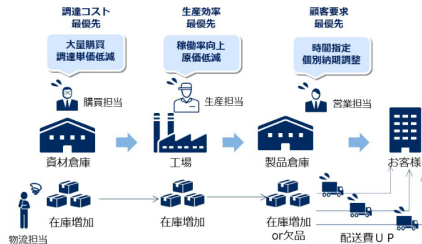


図1 サプライチェーンの問題

1.2. 本研究の目的

3/18

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

ブルウィップ効果には多くの複合的な要因が影響している。そこで、複合的な要因を考慮した効果を明らかにする。また、要因が沢山あるので主効果だけではなくて交互作用がある。組み合わせ爆発が起こるので並列分散処理が必要である。
本研究では、シミュレータを開発し複合要因の影響度合いを明確化する。そして、ブルウィップ効果の低減方法を提案する。

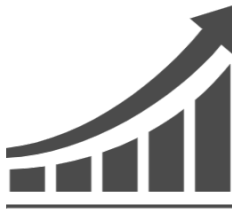


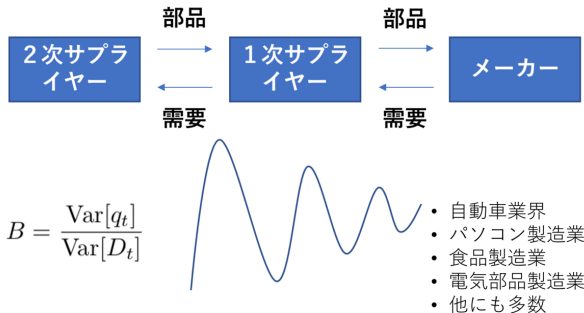
図2 組み合わせ爆発



図3 並列コンピュータ

2.1 サプライチェーンにおけるブルウィップ効果

4/18



需要に鞭のようなバラツキが発生

図 4 サプライチェーンとブルウィップ効果

B : ブルウィップ効果、 q_t : 発注量、 D_t : 需要量
 $\text{Var}[q_t]$: 発注量の分散、 $\text{Var}[D_t]$: 需要量の分散

D. Simchi-Levi, P.Kaminsky and E. Simchi-Levi: Designing and Managing the Supply Chain Concepts, Strategies, and Case Studies, McGraw Hill (2000)

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

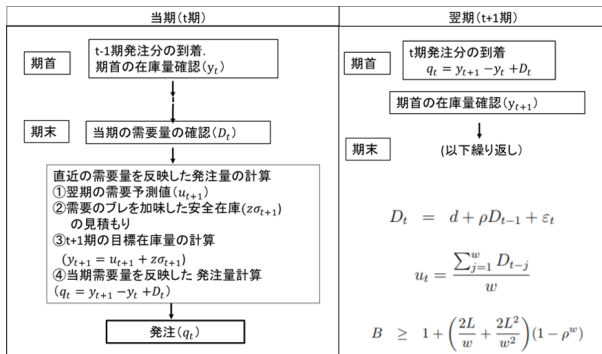


図 5 小売業の発注方法

ρ : 前期の需要量との相関を表すパラメータ, L : リードタイム
 w : 移動平均における過去のデータの採取数, ε_t : 需要予測値の誤差
 z : 安全在庫係数, σ_t : t 期における需要予測誤差の標準偏差の推定量
 d : 需要量の平均

内示生産システムにおけるブルウィップ効果

6/18

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

	M0				M1				M2				M3			
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
月次内示			▽ 金	▽ 金	<div>日別/週別/月別(≒M4)</div> <div>※直近は日量。期間によって週量、月量となる</div> <div>日別/週別(≒M2)</div> <div>※直近は日量。期間によって週量となる</div>											
週次内示			▽ 金	▽ 金	■											
納入指示(日)			▽ 金	■	3日前先1日確定											
			▽ 金	■	3日前先1日確定											

図 6 内示生産システムの発注方法

$$B_w = \frac{M + z^2 \text{Var}[(\hat{\sigma}_{t+1} - \hat{\sigma}_t)]}{M} \quad B_m = \frac{2M + N + z^2 \text{Var}[(\bar{\sigma}_{t+2} - \hat{\sigma}_{t+1} - \hat{\sigma}_t)]}{M}$$

M : t 期における週次内示と確定注文のブレの分散,

N : t 期における週次内示と月次内示のブレの分散,

$\hat{\sigma}_t$: t 期における週次内示と確定注文のブレの標準偏差の推定量,

$\bar{\sigma}_t$: t 期における週次内示と月次内示のブレの標準偏差の推定量

2.2. ブルウィップ効果に影響する要因

7/18

実際の週次内示 \hat{D}_t と確定注文 D_t との差

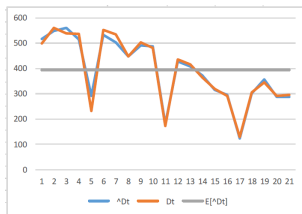


図7 $\hat{D}_t, D_t, E[\hat{D}_t]$ のグラフ

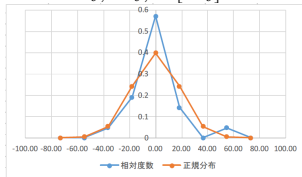


図9 $\hat{D}_t - D_t$ の相対度数分布

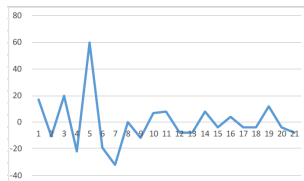


図8 $\hat{D}_t - D_t$ のグラフ

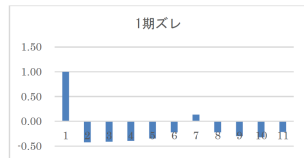
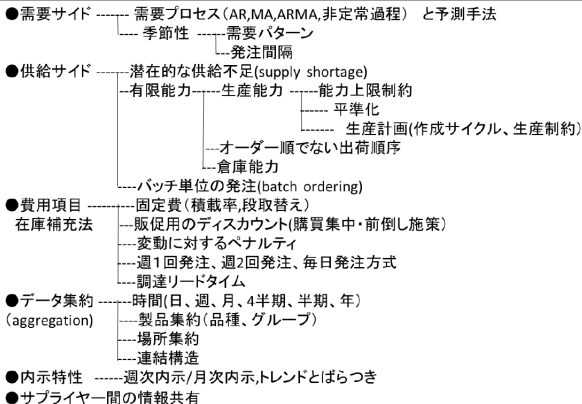


図10 $\hat{D}_t - D_t$ の自己相関係数

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

<合理的な意思決定による場合>



<非合理的な人間判断による場合>

- 合理性追求欠如
- 恣意的な合理的結果の変更 需要ピークを外した発注
- 需要の変化に対するオーバーアクション、品切れに対する過剰な挽回策

図 11 ブルウィップ効果の実際的な要因

本研究で考慮する要因

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

- 1 内示のトレンドの変動 (水準数:27) ←理論でも考慮
- 2 内示のばらつきの変動 (水準数:5) ←理論でも考慮
- 3 発注量のバッチサイズ (水準数:5+10) ←理論でも考慮
- 4 能力上限制約 (水準数:6) ←理論でも考慮
- 5 発注から納入までのリードタイム (水準数:7)
- 6 需要のばらつき (水準数:5)
- 7 安全な在庫目標の決め方 (水準数:5)

- 1 2 要因 (i, j) の組み合わせが $6C2=15$
- 2 2 要因以外の特定の組み合わせに対して 1000 回
- 3 期間は 400 日間 $y_{ij1} \sim y_{ij1000}$

3.1. 数理モデルによるブルウィップ効果の把握

10/18

表 1 要因による週次内示のブルウィップ効果

ケース	考慮する項目			内示生産システムのブルウィップ効果の式 (週次内示の場合)	ブルウィップ効果への影響
	需要 プロセス	設備 能力 制約	バッチ オー ダー		
① ・ $y_{t+1}-y_t=0$ ・設備能力制約なし ・バッチオーダーなし	○	×	×	1	ブルウィップ効果なし
② ・ $y_{t+1}-y_t \neq 0$ ・設備能力制約なし ・バッチオーダーなし	○	×	×	$\frac{M + \text{Var}[y_{t+1} - y_t]}{M}$	在庫目標の変動により増幅
③ ・ $y_{t+1}-y_t \neq 0$ ・設備能力制約あり ・バッチオーダーなし	○	○	×	$\frac{M + \text{Var}[y_{t+1} - y_t]}{M} \cdot \frac{\text{Var}[\hat{Q}_t]}{\text{Var}[Q_t]}$	在庫目標の変動により増幅、 設備能力制約により減幅の複合化
④ ・ $y_{t+1}-y_t \neq 0$ ・設備能力制約なし ・バッチオーダーあり	○	×	○	$\frac{M + \text{Var}[y_{t+1} - y_t]}{M} \cdot \left(1 + \frac{E[X_t(OS - X_t)]}{\text{Var}[\hat{Q}_t]}\right)$	在庫目標の変動により増幅 バッチサイズにより増幅の複合化
⑤ ・ $y_{t+1}-y_t \neq 0$ ・設備能力制約あり ・バッチオーダーあり	○	○	○	$\frac{M + \text{Var}[y_{t+1} - y_t]}{M} \cdot \frac{\text{Var}[\hat{Q}_t]}{\text{Var}[Q_t]} \cdot \left(1 + \frac{E[X_t(OS - X_t)]}{\text{Var}[\hat{Q}_t]}\right)$	在庫目標の変動により増幅 設備能力制約により減幅 バッチサイズにより増幅の複合化

y_t : 在庫目標, Q_t : 発注量

\hat{Q}_t : 設備能力制約ありの時の発注量, OS : バッチサイズ

X_t : Q_t を OS で割った余り

ブルウィップ効果への交互作用 (在庫目標の変動・バッチサイズ)

11/18

④	<ul style="list-style-type: none"> ・$y_{t+1} - y_t \neq 0$ ・設備能力制約なし ・バッチオーダーあり 	○	×	○	$\frac{M + \text{Var}[y_{t+1} - y_t]}{M} \cdot \left(1 + \frac{E[X_t(OS - X_t)]}{\text{Var}[\hat{Q}_t]} \right)$	在庫目標の変動により増幅 バッチサイズにより増幅の複合化
---	---	---	---	---	---	---------------------------------

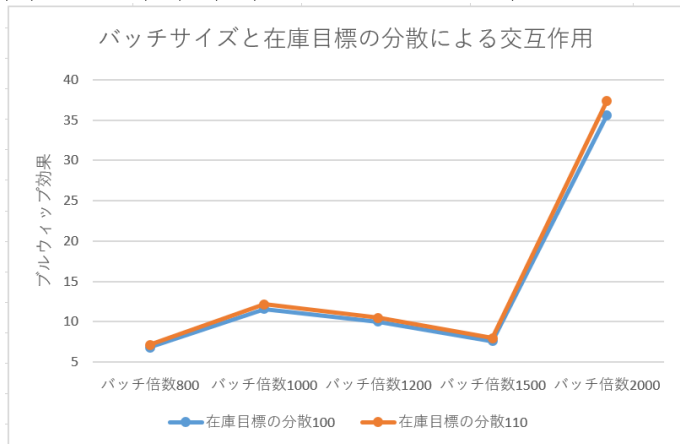


図 12 数理モデルによる交互作用

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

3.2. シミュレータによる交互作用の把握

12/18

主効果と交互作用

主効果とは、それぞれの独立変数がそれぞれ「独自」に従属変数へ与える単純効果のこと。交互作用とは、独立変数を組み合わせた場合の複合効果のこと。

表 2 二元配置の実験表

	B_1	B_2	\cdots	B_b
A_1	y_{111}	y_{121}	\cdots	y_{1b1}
	\vdots	\vdots	\cdots	\vdots
	y_{11k}	y_{12k}	\cdots	y_{1bk}
A_2	y_{211}	y_{221}	\cdots	y_{2b1}
	\vdots	\vdots	\cdots	\vdots
	y_{21k}	y_{22k}	\cdots	y_{2bk}

A_1, A_2, B_1, B_2, B_3 : 要因 A, B の水準

y_{111}, \cdots, y_{2bk} : 結果のデータ (自分の研究ではブルウィップ効果)

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

シミュレータにより複数の要因から交互作用を導出するために実験計画法を活用する。

多元配置法 因子が $r \geq 3$ の場合は, (6), (7) を必要なだけ繰り返す.

- (1) 二つの因子の組合わせ全てについて, 二元配置実験表を作成する.
- (2) 修正項 CT , 総二乗和 S_{TE} を求める. 自由度 f_{TE} は総データ数 - 1 である.

$$CT = \frac{(\text{総計})^2}{\text{総データ数}}, \quad S_{TE} = \sum (\text{個々のデータ値})^2 - CT$$

- (3) 1 因子の主効果 (例えば A) の二乗和 S_A を求める. 自由度 f_A は水準数 - 1 である.

$$S_A = \sum_{i=1}^a \frac{(A_i \text{で測定されたデータの合計})^2}{A_i \text{で測定されたデータ数}} - CT$$

- (4) 2 因子の組合わせ効果 (例えば AB) の二乗和 S_{AB} を求める.

$$S_{AB} = \sum_{i=1}^a \sum_{j=1}^b \frac{\{(A_i, B_j) \text{で測定されたデータの合計}\}^2}{(A_i, B_j) \text{で測定されたデータ数}} - CT$$

- (5) 2 因子の交互作用 (例えば $A \times B$) の二乗和 $S_{A \times B}$ を求める. 自由度 $f_{A \times B}$ は $f_A \times f_B$ である.

$$S_{A \times B} = S_{AB} - S_A - S_B$$

- (6) 3 因子の組合わせ効果 (例えば ABC) の二乗和 S_{ABC} を求める.

$$S_{ABC} = \sum_{i=1}^a \sum_{j=1}^b \sum_{k=1}^c \frac{\{(A_i, B_j, C_k) \text{ で測定されたデータの合計}\}^2}{(A_i, B_j, C_k) \text{ で測定されたデータ数}} - CT$$

- (7) 3 因子の交互作用 (例えば $A \times B \times C$) の二乗和 $S_{A \times B \times C}$ を求める. 自由度 $f_{A \times B \times C}$ は $f_A \times f_B \times f_C$ である.

$$S_{A \times B \times C} = S_{ABC} - S_A - S_B - S_C - S_{A \times B} - S_{B \times C} - S_{C \times A}$$

- (8) 残差二乗和 S_E を S_{TE} から各要因の変動を全て取り去ることで求める. 自由度 f_E は f_{TE} から各要因の自由度を全て取り去ることで求める.

3.3. Message Passing Interface(MPI) による 並列分散処理

15/18

MPI

MPI とは、並列コンピューティング利用するための標準化された規格である。**MPI** はプログラミング言語とは独立の通信プロトコルで、並列計算機上で動くプログラムに使用される。

実験環境



図 13 並列計算機

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

GPGPU

グラフィックス処理以外の演算を **GPU** に行わせるようにしたもの。
CUDA、AMD Stream、OpenCL、DirectCompute 等がある。

表5 CUDAによるGPGPU化の特徴

観点	特徴	評価
開発環境	NVIDIA 社製の GPU が搭載されており、VisualStudio がインストールされていることを前提として、NVIDIA 社から提供されているインストーラを用いて、統合開発環境をインストール可能である。	○
開発効率	独自言語 CUDA を新たに覚える必要がある。使用する GPU の特性を考慮し、ソフトウェアを再設計する必要があるため、GPU コードの作成に時間を要する。	△
性能	GPU の特性を考慮したチューニングが可能なため、効果的・効率的に GPU を使用できる。	◎
互換性	NVIDIA 社が製造する GPU 以外では動作しない。	×

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

実験方法

配列の加算処理の比較

```
(base) hiramatsu@zeta-desktop:~/cuda-samples/Common$ nvcc main2.cu -o main2
(base) hiramatsu@zeta-desktop:~/cuda-samples/Common$ ./main2
./main2 開始...
使用するデバイス 0: GeForce RTX 2070 SUPER
配列サイズ: nx 12192 ny 12192
配列初期化処理      3.560431 秒
配列加算処理(CPU) 0.301053 秒
配列加算処理(GPU) 0.004645 秒
配列が一致します。
```

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに

4. まとめと今後の課題

18/18

まとめ

- 1 **CUDA** のプログラミング

今後の課題

- 1 **MPI** を用いたシミュレーター完成
- 2 それを用いて実験
- 3 **CUDA** を用いたシミュレーター作成

1. はじめに
2. 従来手法
3. 提案手法
4. おわりに